Prototipo de software para el reconocimiento facial por medio de visión artificial usando la metodología de análisis EigenFaces

Juan Diego Hoyos Monroy
2019.

Universidad Tecnológica de Pereira.
Ingeniería de Sistemas y Computación.

Dedicatoria

A mi familia, en especial mi madre y mi tía Rosalia por brindarme todo su apoyo incondicional, por su comprensión y su amor, a la universidad por brindarme la oportunidad de mejorar como ser humano brindándome todas las herramientas académicas y a todas esas personas qué aportaron de forma desinteresada en el mejoramiento de mi como ser humano y profesional.

Agradecimientos

A mi familia quién fue la motivación en todo momento y el apoyo incondicional sin el cual pudiese habido alcanzado estos logros.

A mi director de proyecto el PhD Guillermo Roberto Solarte por brindarme su tiempo, orientación y motivación.

A la Universidad Tecnológica de Pereira junto con su programa de Ingeniería de Sistemas y Computación por hacer posible este proyecto y la culminación de mi carrera profesional.

A todos los docentes y amigos cuya calidad humana me permitió llegar más allá de lo académico y llegar a mejorar día a día como ser humano.

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

PhD. Guillermo Roberto Solarte Martinez

_____

Tabla de Contenidos

Lista de figuras

Abstract

En presente trabajo usted encuentra el desarrollo, análisis y explicación de un prototipo de software para la identificación y reconocimiento facial por medio de visión artificial, algoritmos y metodologías haciéndolo de una forma optima y eficiente. Se describe paso a paso como funciona los algoritmos elegidos para dicha labor; como se realiza cada uno de los análisis y como se mitigan cada uno de los problemas que llegan a afectar la eficiencia y eficacia de los algoritmos. También se describe como se extraen los componentes principales de las imágenes, como identificarlos y cuales se toman además, de como a través de los años se ha intentado llegar a un modelo robusto y eficaz; capaz de combatir las adversidades de calidad y condiciones lumínicas y lograr el cometido de identificar un rostro.

Capítulo 1

Generalidades.

Introducción

A comienzos del 2018 el desarrollo de software ha ido tomando un rumbo hacia la automatización completa de procesos qué usualmente requerían de un ser humano para la verificación de la información. Las empresas más influyentes y exitosas de esta época han demostrado su gran avance tecnológico y su visión acerca de la utilidad y potencial de las nuevas tecnologías algorítmicas y de software. El reconocido inversor Elon Musk afirmó lo siguiente: Elon Musk. (jul. 2017*). "Ciertamente habrá una interrupción laboral. Porque lo que va a suceder es que los robots puedan hacer todo mejor que nosotros. Me refiero a todos nosotros"*, dijo Musk, en declaraciones a la National Governors Association; refiriéndose a él gran avance qué ha tenido el aprendizaje automático en robótica y demás aplicaciones.

Muestra de ello es el reconocimiento facial. La reconocida empresa Facebook posee esta tecnología sobre sus usuarios, de esta forma sugiere etiquetados en fotografías y evita suplantaciones de usuarios. Claramente el aprendizaje automático y la visión artificial han ido de la mano para alcanzar los sistemas que actualmente se conocen, como lo mencionó Montse Hidalgo. (2018). "*Los sistemas de reconocimiento de imagen, que nunca habían sido tan inteligentes, amplían cada día sus capacidades de detección e identificación de objetos"*.

Este prototipo de software se encargará de fusionar ambas tecnologías. Se llevará la tecnología del reconocimiento facial hacia un enfoque diferente, ofreciendo el prototipo de software como una solución funcional y económica. Logrando así abrir las puertas de esta tecnología a cualquier usuario y/o persona que desee aprovechar este prototipo a su beneficio.

Planteamiento del problema

Actualmente en 2018 se disponen de tecnologías orientadas a la automatización de la autenticación de nosotros como seres humanos en el ámbito social. Dichas tecnologías requieren de un despliegue de hardware y software de gran robustez, lo que limita su acceso a todo tipo de usuario.

Una de ellas es el reconocimiento facial, la cual posee de biométricas altamente eficientes pero que por cuestiones de hardware y software muchas personas han desistido acerca de usarlo. Dicho esto, los nuevos avances y accesibilidad a los algoritmos, la publicación de librerías de visión artificial y aprendizaje automático; han permitido lograr alcanzar prototipos de software de reconocimiento facial que suplen las soluciones costosas por una solución funcional, económica y eficiente.

Delimitación

En este trabajo se mostrará un prototipo de software para el reconocimiento y detección de rostros. Todo esto, por medio de algoritmos y métodos que se describirán de forma especifica en las siguientes paginas.

Este prototipo estará única y exclusivamente limitado a ser evaluado en el ambiente académico de la Universidad Tecnológica de Pereira y se hará con fines evaluativos respecto a desempeño, eficiencia y costo sobre los métodos y algoritmos usados.

Objetivos

Objetivo General

Desarrollar un prototipo de software para el reconocimiento facial por medio de visión artificial usando la metodología de análisis EigenFaces

Objetivos específicos

- Análisis y diseño para el prototipo de software.
- Construcción en desarrollo del prototipo.
- Desarrollar y ejecutar plan de pruebas al prototipo de software.
- Elaborar documentación de los resultados de las pruebas.

Diseño metodológico

La ingeniería de software nos provee un enfoque sistemático, disciplinado y cuantificable permitiéndonos llevar a cabo numerosas tareas agrupadas de distintas formas; estructurada y documentada. Ahora bien, en 2019 las metodologías de desarrollo ágil de software han tomado fuerza en gran parte gracias al crecimiento tecnológico y a la alta demanda de aplicaciones en poco tiempo. En los siguientes fragmentos se describirá el proceso metodológico hibrido que se realizo entre Desarrollo adaptativo de software (ASD) y la ingeniería de software para obtener los mejores resultados en tiempo, calidad y costo. Primero se obtuvieron los requisitos funcionales o no funcionales de la aplicación esto permite dar la visión acerca de que se encargara de hacer el prototipo y de forma documentada y organizada ver que funcionalidades fundamentales incorporara. Esto se logra por medio de la especificación de los requisitos para lograr una clara interpretación de los comportamientos esperados en el prototipo; una vez desarrollado. Cada uno de los requisitos y el comportamiento se analizarán por medio de herramientas como Casos de uso, historias de usuario, diagrama de clases, diagrama de despliegue.

Capítulo 2

Marco teórico.

Detección de rostros

Es un caso especifico en la detección de objetos. Es un proceso qué se encargará de ubicar los rostros en el video o imagen qué se esté analizando; siendo el mismo un proceso no tan sencillo como lo haría el sistema de visión humana. En este proyecto se usa el algoritmo Haar Cascade.

Haar Cascade

La detección de objetos usando los clasificadores basados en Haar Cascade es un método efectivo de detección de objetos propuesta por Paul Viola y Michael Jones en su Paper "Rapid Object Detection using a Boosted Cascade of Simple Features" en 2001.

Inicialmente, el algoritmo necesita un número grande de imágenes positivas y negativas, es decir un numero de imágenes con rostros y otra sin rostros para entrenar nuestro clasificador. Una vez hecho esto necesitaremos extraer las características de el; Para esto se usan las características de Haar que se muestran en la siguiente página (Figura 1. Características Haar). Estas son una especie de núcleo convolucional. Cada característica es un valor único objetivo de restar la suma de pixeles debajo del rectángulo blanco de la suma de pixeles debajo del rectángulo negro.

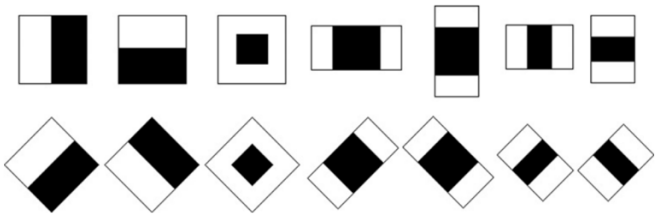

Figura 1. Características Haar. Ref [4] [Figura 8.21].

Ahora bien, cada ubicación y tamaño posible de los núcleos se usa para calcular muchas características, pero esto se traduce directamente en un problema de computo. Para resolver esto, los autores introdujeron el concepto de imagen integral; lo que traduce que no importa cuán grande sea la imagen, los cálculos que se hace para cada píxel dado a una operación solo involucra 4 pixeles. Lo que mejora considerablemente la optimización y velocidad del procesamiento. Sin embargo, se nota que entre todas esas características hay muchas que son irrelevantes. Por ejemplo, observemos la Figura 2 al final de la página 7. La fila superior muestra dos buenas características. La primera característica seleccionada parece centrarse en la propiedad de que la región de los ojos suele ser mas oscura que la región de la nariz y las mejillas, la segunda se basa en que los ojos son mas oscuros que el puente de la nariz como lo podemos ver en la figura 2.

Figura 2. Características Haar relevantes e irrelevantes. Ref [4] [Figura 8.22].

Analizando esto, la pregunta que surge es, ¿Como lograr seleccionar las mejores características de las 160000+ posibles?, Bueno para ello se usa Adaboost, aplicamos

cada característica en todas nuestras imágenes de entrenamiento. Para cada característica, esta acción encontrara el mejor umbral que clasificara un rostro o cara como positiva y negativa. Claramente, esta forma obtendrá errores y clasificaciones erróneas. Por ello, seleccionaremos las características con la menor tasa de error, esto significará que son las características mas precisas para clasificar caras y lo que no sea una cara. Realmente el proceso es mas complejo, cada imagen se le asigna un peso en el inicio. Después de cada clasificación, los pesos de las imágenes mal clasificadas son incrementados. Entonces el mismo proceso se realiza; obteniendo nuevas tasas de error y nuevos pesos. El proceso se sigue realizando hasta que se logre la precisión requerida o la tasa de error encuentre el numero de características requerido.

Se denomina pequeño por que el clasificador por si solo no puede clasificar la imagen, pero juntos logran formar un clasificador fuerte. El Paper mencionado en la página 6 dice que, incluso con 200 características provee una detección con el 95% de precisión y en su configuración final tenían alrededor de 6000 características (De 160000+ a 6000 características).

Reconocimiento de rostros

La tarea de reconocer rostros resulta ser una tarea sencilla para los humanos. Algunos experimentos como por ejemplo el mostrado en el libro: Chiara Turati, Viola Macchi Cassia, F. S., and Leo, I. Newborns face recognition: Role of inner and outer facial features. Child Developvidment 77, 2 (2006), 297–311 demuestra qué hasta los niños de poca edad de nacidos están en la capacidad de distinguir rostros conocidos. ¿Que tan difícil puede ser reconocer un rostro para un computador? Esto saca todo lo poco que sabemos acerca del reconocimiento humano hasta ahora. ¿Se usarán características internas como ojos, nariz y boca?, o por el contrario ¿usaremos características externas como la forma de la cabeza, el cabello para un reconocimiento exitoso? ¿Como el cerebro codifica y analiza las imágenes? Esto fue mostrado por David Husbel y Torsten Wiesel, nuestro cerebro tiene nervios especializados para responder a unas características locales de una escena como líneas, bordes, ángulos o movimiento. Dado a que no apreciamos el mundo como piezas dispersas nuestra corteza visual debe combinar de alguna forma las diferentes fuentes de información en patrones útiles. De ahí la razón por la cual podemos reconocer a una persona por su forma de caminar, su posición al estar de pie y por muchos factores que implícitamente nuestro cerebro nos hace distinguir con gran claridad.

El reconocimiento facial basado en características geométricas de la cara es probablemente el enfoque más intuitivo para el reconocimiento facial. En este trabajo se trabajará sobre la metodología Eigenfaces.

Eigenfaces

Esta metodología toma un enfoque holístico hacia el reconocimiento facial: Una imagen facial es un punto de un espacio de dimensional alto en cuanto a espacio de imagen y una representación de menor dimensión encontrada; donde la clasificación se vuelve más sencilla. El subespacio de dimensión inferior se encuentra con el Análisis de componentes principales (PCA en ingles) que se encarga de identificar los ejes con la varianza máxima. Si bien, este tipo de transformación es optima desde el punto de vista de la reconstrucción, no tiene en cuenta ninguna etiqueta de clase. Imagine una situación en la que la varianza que se genere sea a partir de fuentes externas, sea ligero. Los ejes con varianza máxima no necesariamente contienen ninguna información discriminatoria, por lo que una clasificación se volvería computacionalmente imposible. Por lo tanto, se aplica una proyección especifica de clase con un Análisis Discriminante Lineal al reconocimiento facial en el método Fisherfaces, pero este no será tema de este trabajo de grado.

Volviendo al tema de Eigenfaces el problema con la representación de la imagen como anteriormente se menciono es que las imágenes dadas son dimensionalmente altas. Por ejemplo, supongamos un conjunto de imágenes bidimensionales **pxq** que en escala de grises ocuparan **pq** dimensiones de espacio vectorial, entones una imagen con 100x100 pixeles será una imagen de 10.000 espacio dimensional. La pregunta sería: ¿Serán todas las dimensiones igual de útiles para nosotros? Entonces, para resolverla y tomar una decisión acertada consideramos entonces, que solo son igual de importantes y útiles si

hay alguna variación en los datos, por lo que se buscan los componentes que representan la mayor parte de la información. El análisis de componentes principales (PCA en ingles) como anteriormente se mencionó y que será descrito en este trabajo de grado en los siguientes capítulos se encargará de encontrar las direcciones con la mayor variación en los datos y que se les dará el nombre de componentes principales.

Descripción del algoritmo

Consideremos el vector $x = X_1, X_2, X_3, \ldots . X_n$

1. Calculamos la media μ

$$\mu = \frac{1}{n}\sum_{i=1}^{n} X_i$$

2. Calculamos la covarianza de la matriz S

$$S = \frac{1}{n}\sum_{i=1}^{n} (X_i - \mu)(X_i - \mu)^{T_i}$$

3. Calculamos los eigenvalues λ_i y los eigenvectors v_i de S

$$Sv_i = \lambda_i v_i, i = 1,2, \ldots . , n$$

4. Ordene los eigenvectors descendientemente de acuerdo con su eigenvalue. Las k principales componentes son los eigenvectors que corresponden a el eigenvalues mas grande k

   Las k principales componentes observadas del vector X son dadas por

$$y = W^T(x - \mu) \quad \text{, Donde } W = (v_1, v_2, v_3, \ldots, v_{k)}$$

El método Eigenfaces realiza un reconocimiento facial por:

- Proyección de todos los datos de ejemplo dentro de un el subespacio PCA
- Proyección de una imagen consultada dentro del subespacio PCA
- Encontrando el vecino mas cercano entre los datos de entrenamiento proyectados y la proyección de la imagen de consulta.

por Karl Pearson (1901) y Harold Hotelling (1933) para convertir variables posiblemente correlacionadas en un conjunto mas compacto y pequeño

En inglés PCA (Principal Component Analysis). Es una técnica popular de reducción dimensional en aprendizaje automático de máquinas.

Análisis de componentes principales (ACP)

En inglés PCA (Principal Component Analysis). Es un procedimiento estadístico que extrae las características mas importantes de un conjunto de datos. Consideraremos un conjunto de datos mostrados en la figura 1.

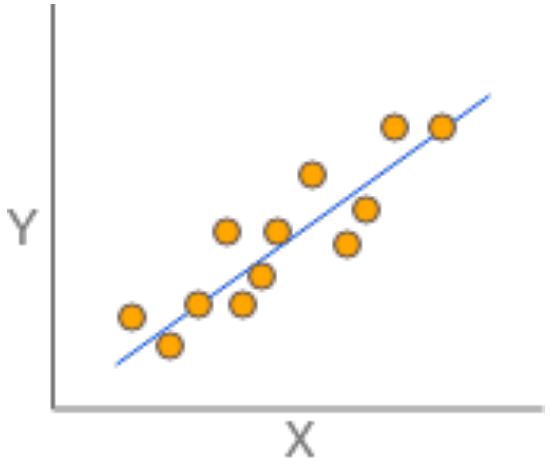

Figura 3. PCA Puntos arbitrarios en 2 dimensiones.

Considerando los puntos arbitrarios en 2 dimensiones mostrados anteriormente, cada dimensión corresponde a una característica en la cual se esta interesado. Aquí algunos argumentarían que los puntos se establecen en un orden aleatorio; Sin embargo, si ampliamos la perspectiva un poco se notara un patrón lineal (indicado por la línea azul). Un punto clave de PCA es la reducción de dimensional. La reducción de dimensional es el proceso de reducir un número de dimensiones de un conjunto de datos; por ejemplo, en la figura de abajo es posible aproximar el conjunto de puntos en una sola línea y por lo tanto reducir la dimensión de los puntos dados de dos dimensiones a una dimensión.

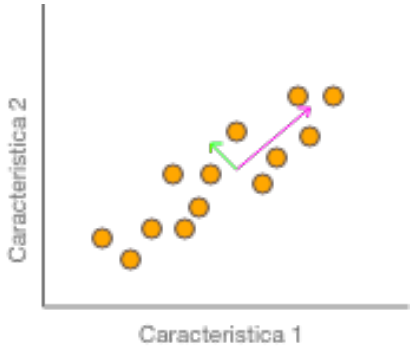

Figura 4. PCA Reduccion dimensional

Además, también se pueden distinguir que los puntos varían mas a lo largo de la línea azul, mas que lo qué varían a lo largo de los ejes característica 1 o característica 2. Esto significa que si se conoce la posición de un punto a lo largo de la línea azul tiene mas información el punto si supiéramos cuales valores tienen en los ejes característica 1 o característica 2. Por lo tanto, el algoritmo permite encontrar la dirección a lo largo de la cual nuestros datos varían en mayor cantidad; De hecho, el resultado de correr el algoritmo PCA en un conjunto de datos para el ejemplo consiste en 2 vectores llamados vectores propios (eigenvectors, en ingles) que son los componentes principales del conjunto de datos.

El tamaño de cada vector propio (eigenvector, en ingles) se codifica en el valor propio (eigenvalue) correspondiente e indica cuando varían los datos a lo largo del componente principal. El comienzo de los eigenvectors es el centro de los datos, la aplicación del algoritmo al conjunto de datos de N-dimensiones produce un numero de eigenvectores de N-dimensiones, un numero de eigenvalues N y un punto central de 1N dimensiones.

Descripción del calculo PCA

La meta es transformar un conjunto de datos X de una dimensión p en un conjunto de datos alternativo Y de menor dimensión L. De manera equivalente, buscamos encontrar la matriz Y; donde Y, es la Karhunen–Loève transform (KLT) de la matriz X.

$$Y = KLT\{X\}$$

Ahora supongamos que el conjunto de datos comprende un conjunto de observaciones de las variables p, y desea reducir los datos para que cada observación puede describirse con solo L variables (L < p). Supongamos, además que los datos se organizan como un conjunto de n vectores de datos x1, …, xn; donde cada xi representa una observación agrupada única de las variables p.

- Escribir $x_{1,}x_{2,}x_{3,}\ldots,x_n$ como vectores en filas, cada uno con p columnas.
- Pon una fila de vectores en una sola matriz **X** de dimensiones **n x p**

Ahora calculamos el promedio de forma empírica

- Encontrar el promedio de forma empírica a lo largo de cada dimensión j = 1,2,3, …, p
- Pon el promedio calculado de valores en un vector de promedios empíricos **u** de dimensiones **p x 1**.

$$u[j] = \frac{1}{n}\sum_{i=1}^{n} x[i,j]$$

El siguiente paso es calcular las desviaciones de el promedio

$$B = X - hu^T$$

Donde **h** es un vector de n x 1 de todos los 1s. Expresado como:

$$h[i] = 1, i = 1,2,3, \dots, n$$

Finalmente, previo a hallar los eigenvectors y las eigenvalues encontraremos la covarianza

$$C = \frac{1}{n-1} B^*.B$$

Donde * es el operador de trasposición conjugado. Debemos tener en cuenta que, si **B** se compone completamente de números reales, como ocurre en un sin numero de aplicaciones la trasposición conjugada es la misma que la transposición normal.

Para encontrar los eigenvectors y las eigenvalues calculamos la matriz V la cual es diagonal a la matriz de covarianza C.

$$V^{-1}CV = D$$

Donde D será la matriz diagonal de eigenvalues de C. Ahora, de la matriz D tomaremos la forma de una matriz p x p diagonal a la matriz anteriormente descrita de la siguiente forma:

$$D[k,l] = \begin{cases} \lambda_k, k = 1 \\ 0, k \neq 1 \end{cases}$$

Tenemos que λ_j es el j eigenvalue de la matriz de covarianza C. Los eigenvalues y eigenvectors son ordenados y emparejados.

Capítulo 3

Requerimientos y diseño del sistema.

Análisis de requerimientos

Requerimientos funcionales

RF1. Crear un rostro con información qué lo identifique.

RF2. Eliminar un rostro previamente registrado.

RF3. Listar los rostros registrados.

RF4. Identificar los rostros previamente registrados.

RF5. Identificar similitud a un rostro existente sí el rostro analizado no ha sido previamente registrado.

RF6. Modificar el identificador de algún rostro registrado.

Requerimientos no funcionales

RNF1. La aplicación será capaz de identificar un rostro y aproximar su identificación en menos de 10 segundos.

RNF2. El Software podrá ser utilizado en los sistemas operativos Windows, Linux y Mac OS.

RNF3. La aplicación debe poder utilizarse solo instalando la versión del interprete correcta y las librerías para su funcionamiento.

RNF4. La aplicación requiere de una cámara conectada al computador para funcionar.

RNF5. La aplicación no requiere infraestructura de comunicación, conexión a redes ni ningún tipo de cifrado especial en conexiones.

Diseño

Análisis de requerimientos: Casos de uso

Caso de uso: Registrar nueva cara

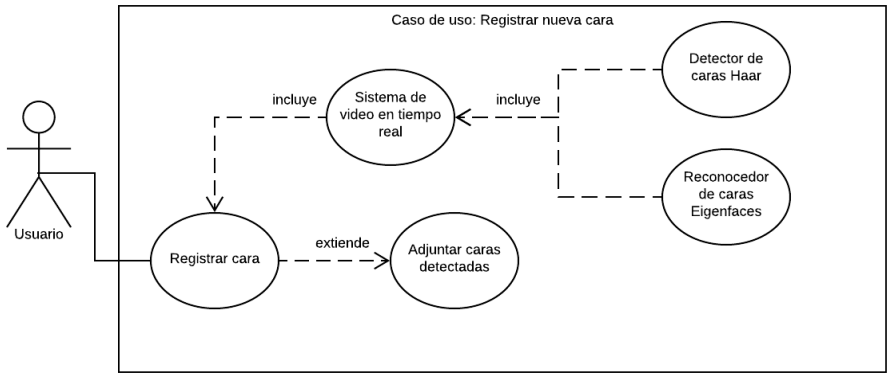

Figura 5. Caso de uso: Registrar una nueva cara

Caso de uso: Listar caras

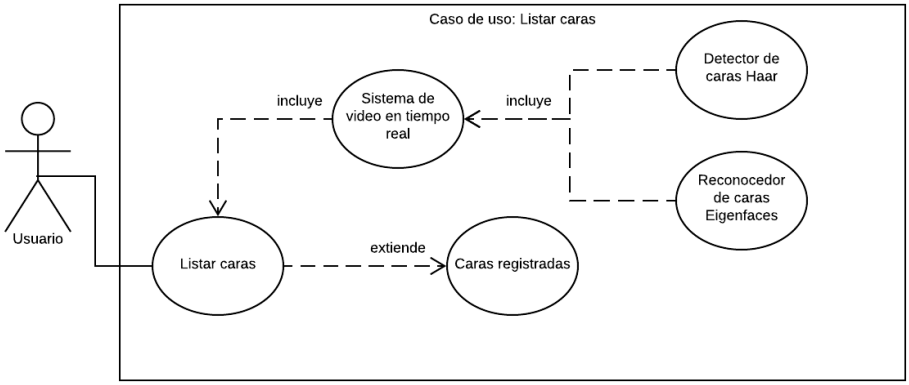

Figura 6. Caso de uso: Listar caras

Caso de uso: Eliminar cara

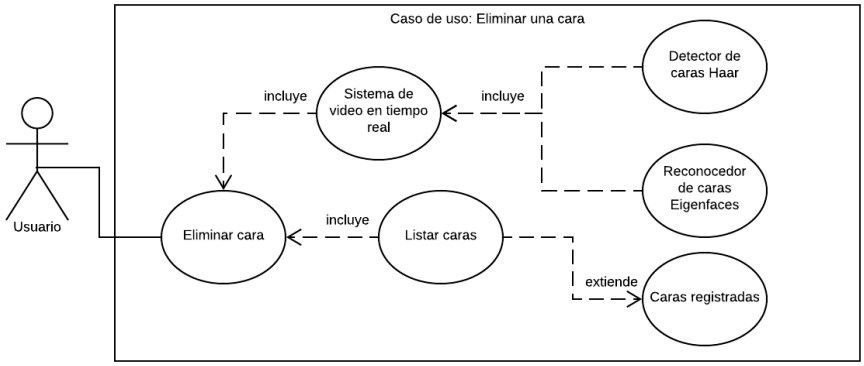

Figura 7. Caso de uso: Eliminar una cara

Descripción del escenario: Diagramas de secuencia

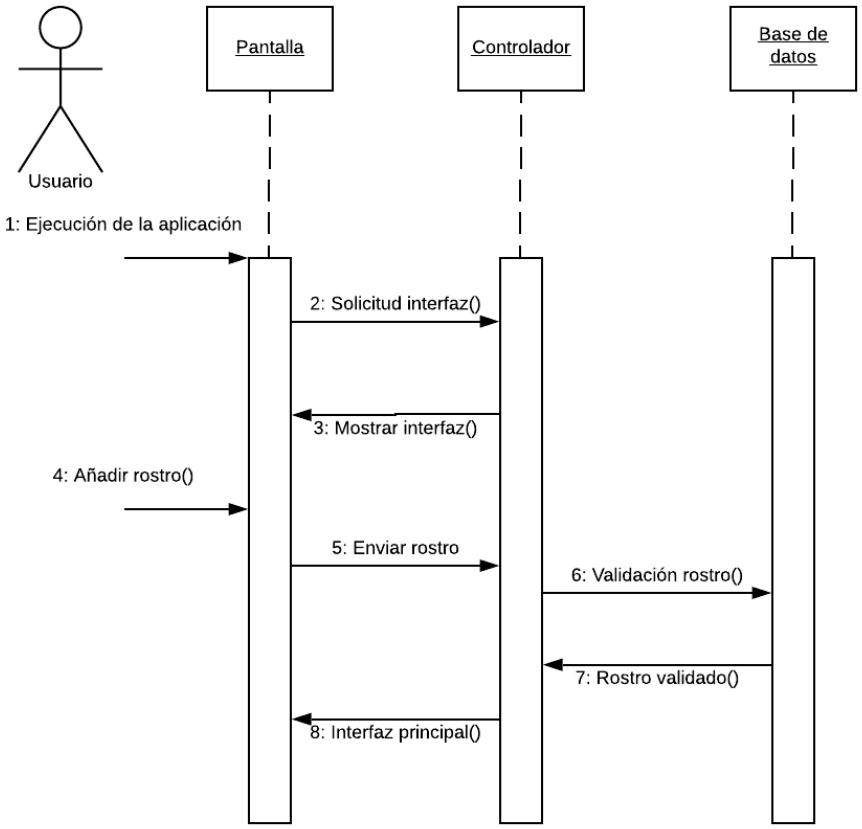

Figura 8. Diagrama de secuencia: Crear rostro

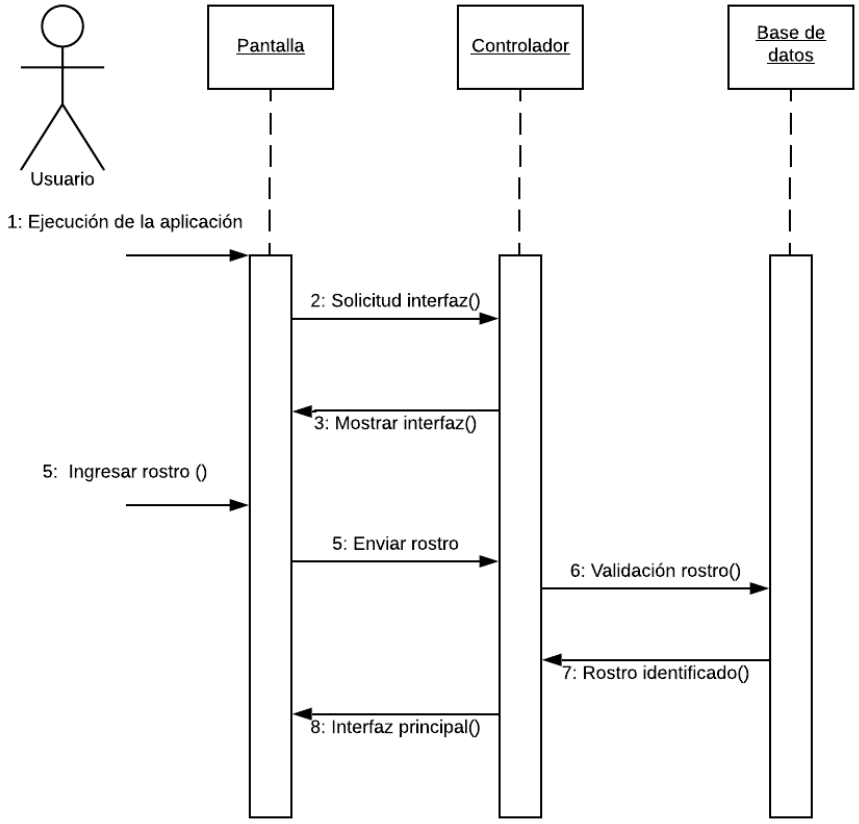

Figura 9. Diagrama de secuencia: Identificar rostro

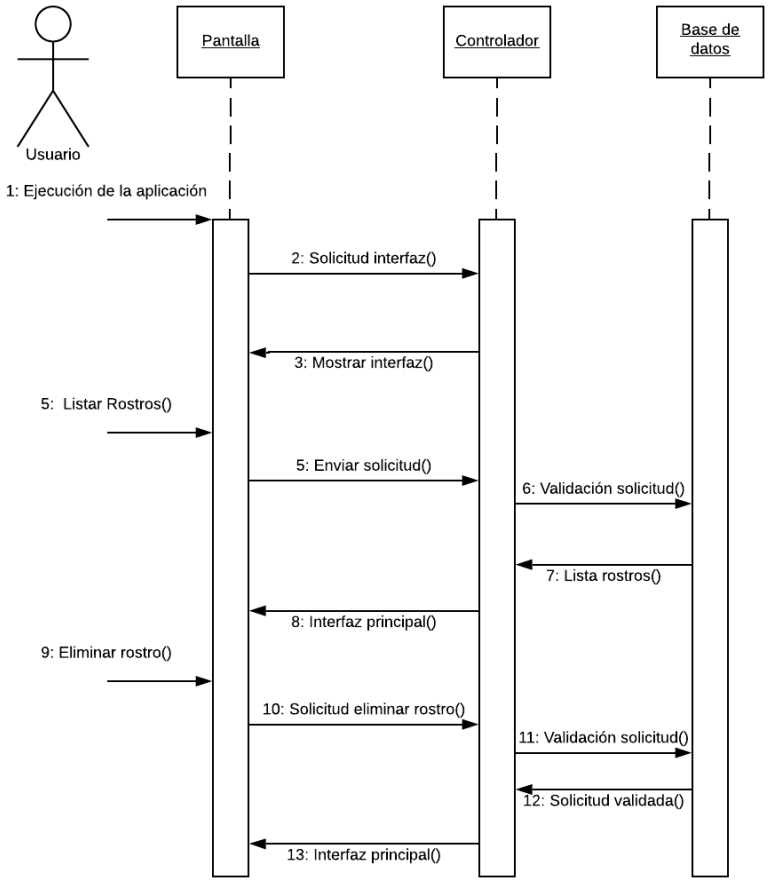

Figura 10. Diagrama de secuencia: Eliminar rostro

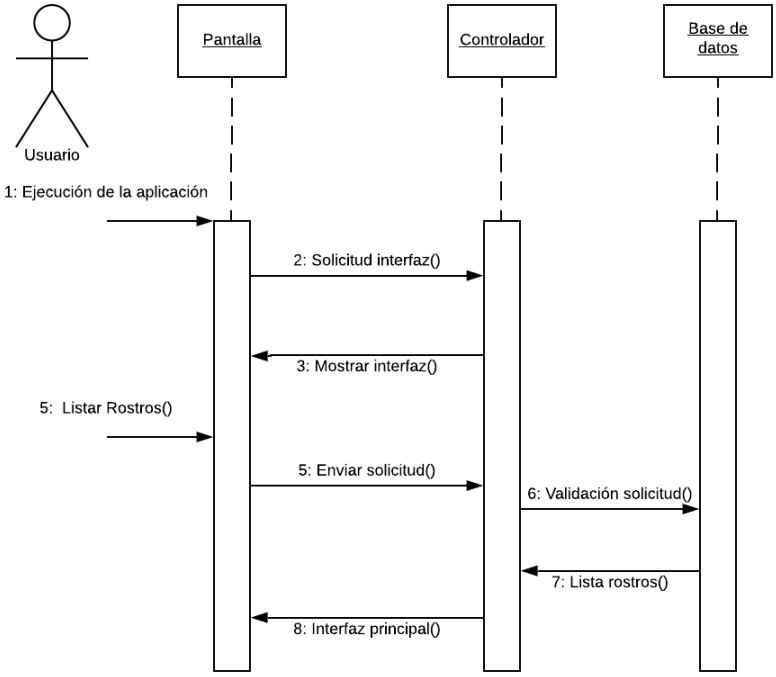

Figura 11. Diagrama de secuencia: Lista rostros

Diagrama de clases

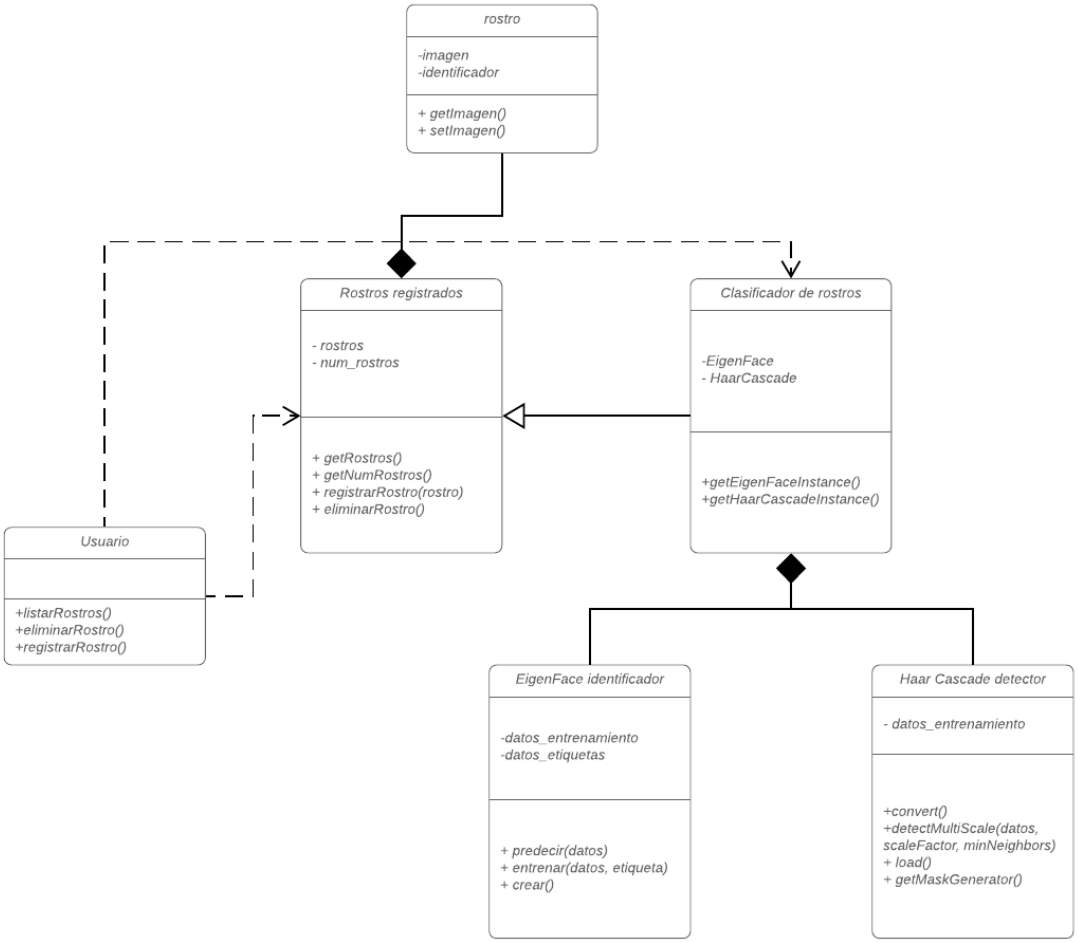

Figura 12. Diagrama de clases

Capítulo 4

Detección e identificación de rostros en código.

La primera consideración que se tiene en cuenta para iniciar con el desarrollo es dividir el aplicativo en módulos específicos que juntos harán un prototipo funcional. En este caso especifico, se divide en módulos de: Captura de video y manipulación de imágenes, Detección de rostros, Identificación de rostros y finalmente las demás funcionalidades como listar, eliminar, y crear que estarán todas dentro de un mismo modulo y qué por efectos de optimización de tiempo y análisis del algoritmo no serán implementados en este prototipo.

Para capturar el video de la cámara configurada por defecto en el sistema operativo se usa la función de la librería OpenCV VideoCapture de la siguiente forma:

```
captura = cv2.VideoCapture(0)
```

Se inicializan los modelos; se crean y se entrenan

```
haar_cascade = cv2.CascadeClassifier('archivo_entrenamiento.xml')
eigenfaces = cv2.face_EigenFaceRecognizer.create()
data = loadCSV().get_train_data()
eigenfaces.train(data)
```

Hecho esto, en la variable captura tenemos todos los datos de la imagen que se está capturando en ese momento. Para desplegarla y mostrarla al usuario, además de poder acceder a los pixeles de forma regular y poder hacer un análisis en tiempo real se

hace un ciclo infinito donde se recaudan los pixeles capturados por la librería para mostrarlos y hacerle una transformación a escala de grises, analizar si se han detectado rostros y si se han detectado enviarlos al modulo de identificación. Todo esto, se la siguiente forma:

```
while 1:
    var1, var2 = captura.read()
    gris = cv2.cvtColor(var2, cv2.COLOR_BGR2GRAY)
    caras_detectadas = haar_cascade.detectMultiScale(gris, 1.3, 5)

  for cara en caras_detectadas:
     cara_gris = cv2.cvtColor(cara, cv2.COLOR_BGR2GRAY)
     prediccion = eigenfaces.predict(normalizer_imagen(cara_gris))
     cv2.putText(var2, prediccion)
  cv2.imgshow('titulo', var2)
```

El objeto eigenfaces se encargará de predecir y de retornar el valor de la cara que reconoce de acuerdo con todos los parámetros y funcionamiento que fue especificado en el capitulo numero 2. Marco teórico de este trabajo.

Capítulo 5

Plan de pruebas.

Alcance de las pruebas

Elementos de las pruebas

Los módulos por probar se especificarán de la siguiente forma:

- Modulo de identificación facial
- Modulo de reconocimiento facial
- Modulo de reconocimiento e identificación en tiempo real
- Modulo de captura de video
- Modulo de creación de los archivos de entrenamiento

Funcionalidades por no probar

Los siguientes módulos no se probarán debido a que su implementación en desarrollo fue omitida debido a la prioridad de dar análisis al algoritmo y su eficiencia:

- Listado de rostros registrados
- Eliminar un rostro registrado
- Registrar un rostro por medio de la interfaz

Enfoque de pruebas

Las pruebas irán enfocadas fundamentalmente a probar la funcionalidad, y el desempeño de los módulos especificados en los elementos de las pruebas. Se omitirá la prueba de interfaces o experiencia de usuario debido a qué el prototipo no tiene implementado una interfaz de usuario completamente funcional.

Criterios de aceptación o rechazo

Los criterios de aceptación serán generales y objetivos para probar netamente funcionalidad y a que cada uno de los módulos cumpla con el comportamiento esperado tal cual fueron diseñados. Los criterios serán:

- El modulo debe seguir los lineamientos del comportamiento esperado
- El modulo debe cumplir con el objetivo para el cual fue diseñado

Recursos

Requerimientos de entornos – Hardware

- Un computador que cumpla las siguientes características:
    - 4 GB RAM o mas
    - Core i3 o mas
- Cámara web conectada al computador.
- Mouse.

- Teclado.
- Monitor.

Requerimientos de entorno – Software

- Python 3.x con las librerías:
    - OpenCV 4.1.0
    - Pandas 0.24.2
    - Pytz 2019.1
    - Six 1.12.0
    - Python-dateutil 2.8.0
    - Numpy 1.16.3
- Sistema operativo Mac OS, Windows o Linux.
- Driver o controlador de la cámara web.

Herramientas de prueba requeridas

- Fotografías de personas
- Personas

Planificación y organización

Procedimientos para las pruebas

Procedimiento general

1. Se ordenarán los datos de entrada y las salidas de acuerdo con lo estipulado en el diseño.
2. Se ingresan los datos al prototipo y se almacenan sus salidas en un archivo separado.
3. Se comparan las salidas esperadas con respecto a las obtenidas realmente
4. Se realiza un análisis de los resultados y se verifica el cumplimiento de la prueba.

Consideraciones con el modulo de reconocimiento facial.

1. Se deberán registrar unas caras previamente.
2. Una vez registradas las caras, se inicia con la construcción del conjunto de datos que se ingresarán al prototipo componiéndola con las caras ya registradas y con otras que no estarán registradas.
3. Se ordenara el conjunto de datos tal cual como se describe en el procedimiento general [Planificación y organización / procedimiento general] y se continúan los demás pasos también descritos en el procedimiento general.

Resultados de las pruebas

Modulo de identificación facial

Tras múltiples pruebas realizadas con diferentes rostros, el prototipo identificó todas las caras; incluso en condiciones lumínicas bajas.

Modulo de reconocimiento facial

El modulo de reconocimiento facial si posee algunos problemas con iluminación baja, aproxima e identifica erróneamente. Por el contrario, en condiciones lumínicas normales (de buena iluminación) se lograr ver la fortaleza del sistema de reconocimiento facial.

También se debe tener en cuenta qué el entrenamiento para un reconocimiento rápido y eficaz debe realizarse por lo menos con 3 imágenes de la cara del sujeto a analizar.

Modulo de reconocimiento e identificación en tiempo real

Funciona bien, su eficacia dependerá directamente del hardware del dispositivo donde se ejecute. Sin embargo, no requiere mucho hardware para realizar los cálculos por lo tanto funciona correctamente.

Modulo de captura de video

Funciona correctamente, captura el video de la cámara por defecto del sistema operativo.

Modulo de creación de los archivos de entrenamiento

Funciona correctamente, crea el archivo csv con sus respectivas fotografías de forma eficiente.

Lista de referencias

[1] Joe Minichino, Joseph Howse (2015). Learning OpenCV 3 Computer Vision with Python - Second Edition. Packt Publishing.

[2] Chiara Turati, Viola Macchi Cassia, F. S., and Leo (2006). Newborns face recognition: Role of inner and outer facial features. Child Development.

[3] Turk, M., and Pentland (1991). Eigenfaces for recognition. Journal of Cognitive Neuroscience.

[4] Kenneth Dawson (2014). A Practical Introduction to Computer Vision with OpenCV (Wiley-IS&T Series in Imaging Science and Technology).

[5] Kieron Messer, Josef Kittler, James ShortG. Heusch, Fabien Cardinaux, Sebastien MarcelYann, Rodriguez, Shiguang ShanY. SuWen GaoX. Chen. (1920). Performance Characterisation of Face Recognition Algorithms and Their Sensitivity to Severe Illumination Changes

[6] Zhao, W., Chellappa, R., Phillips, P., and Rosenfeld, A. (2003) Face recognition: A literature survey. ACM Computing Surveys (CSUR)

[7] Belhumeur, P. N., Hespanha, J., and Kriegman, D. (1997) Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on Pattern Analysis and Machine Intelligence.